A Batch/Recursive Algorithm for 3D Scene Reconstruction

Philip F. McLauchlan,
School of Electrical Engineering, Information Technology and Mathematics,
University of Surrey, Guildford GU2 5XH, UK
Email P.McLauchlan@ee.surrey.ac.uk

Abstract

We present a new formulation of sequential least-squares applied to scene and motion reconstruction from image features. We argue that recursive techniques will become more important both for real-time control applications and also interactive vision applications. The aim is to approximate as well as possible the result of the batch bundle adjustment method. In previously published work we described an algorithm which works well if the same features are visible throughout the sequence. Here we show how to deal with new features in a way that avoids deterioration of the results.

The main theoretical advance here is showing how to adjust the system information matrix when scene/camera parameters are removed from the reconstruction. We show how this procedure affects the sparseness of the information matrix, and thus how to achieve an efficient recursive solution to the reconstruction problem.

1 Introduction

The present work is a long-term solution to the problem of scene and motion reconstruction. Our goal is to develop algorithmic and statistical tools that combine data from multiple images and other sensors. We have previously presented specific applications of our Variable State Dimension Filter (VSDF) method in the areas of 3D reconstruction from image sequences [15, 8], vehicle navigation [14], image mosaicing [13] and camera calibration [9]. The thrust of this work is to develop *general*, *scalable* tools that allow data of multiple types to be incorporated (for instance point *and* line features), different projection models to be used (e.g. projective/affine/perspective camera models), and information from different sensors to be integrated.

Most current approaches to scene reconstruction are batch approaches that apply to a set of images processed offline. A natural and inevitable development will be towards online systems, wherein the results are computed as the data is acquired. This will require recursive algorithms that provide constant computational effort as each new image is added to the sequence. Thus a central feature of our VSDF algorithm is that it runs both in batch mode and recursive mode. The usual manner of running the VSDF algorithm is to apply the batch version over an initial set of images, and then to invoke the recursive mode for processing new data. In recursive mode new scene parameters may be incorporated dynamically, and unwanted parameters discarded; hence the term **V**ariable **S**tate **D**imension **F**ilter. Thus we can handle the introduction of newly visible parts of a scene as the camera/scene moves, and the elimination of obscured or invisible features.

Our work is based on [8], but developed in several ways:

- Parametrised constraints between blocks of the state vector can be incorporated. For instance, surface recovery can be integrated with the feature-based reconstruction [16].
- Incorporation of the standard "recursive partitioning"[1] algorithm from photogrammetry [20]. This is a sparse version of the Gaussian elimination method of solving linear systems.
- A more flexible approach to *gauge freedoms* in the representation, for instance the use of homogeneous quantities in projective reconstruction, and the global coordinate frame ambiguity [11, 10].
- Finally, and the subject of this paper, an improved recursive algorithm. The algorithm can now provide control over the trade-off between accuracy and efficiency, by the use of a time window in the recursive updates. This circumvents problems with the previous method that became apparent when incomplete matching data was used.

In other related work, Broida *et al.* [2] applied the Kalman filter in a straightforward way to the motion and structure estimation problem, using a finite dynamical model. Davison & Murray [5] also applied the Kalman

[1]Here "recursive" means that a large linear system is partitioned so that a smaller system can be solved, which is itself recursively partitioned, etc, NOT in the sense of a recursive filter. Thus we use "recursive partitioning" in both the batch and recursive stages.

filter to estimate landmarks seen from a mobile robot, obtaining the dynamical model from odometry information. These methods share the computational complexity problems mentioned above, taking time cubic with the number of tracked features n. Azarbayejani & Pentland [1] apply the Kalman filter to estimating motion and point depth only. Other methods are faster, sharing with our approach linear complexity with respect to n. Harris & Pike [7] used Kalman filters to estimate each 3D point, with the motion parameters calculated separately. This Euclidean method was extended to projective and affine reconstruction by Beardsley *et al.* [17]. Their method is equivalent to a "simple" form of our algorithm, in which the motion parameters are computed separately, and excluded from the global estimation process. Cui *et al.* [4] describe a batch/recursive algorithm quite closely related to a special case of the current work. Chiuso *et al.* [3] have developed a dynamical systems approach to reconstruction which does not require a batch initialisation. A rather different approach is presented by Fitzgibbon & Zisserman [6], who employ the idea of "motion threading", whereby motion estimates from adjacent image pairs are built up into triplets, then joined into subsequences, using local batch algorithms over the subsequences to reduce the error. In this paper we demonstrate that the accuracy of full batch bundle adjustment can be approached using recursive methods.

2 Batch and Recursive Least-Squares

The Levenberg-Marquardt algorithm [18] is a general modified Gauss-Newton minimization algorithm for the case when derivatives of the objective function can be computed. Let the unknown parameters be represented by the vector $\mathbf{x}$, and let noisy measurements of $\mathbf{x}$ be made:

$$\mathbf{z}(j) = \mathbf{h}(j;\mathbf{x}) + \mathbf{w}(j), \quad j = 1, \dots, k \tag{1}$$

where $\mathbf{h}(j)$ is a measurement function and $\mathbf{w}(j)$ is zero-mean noise with covariance $N(j)$. The maximum likelihood solution for $\mathbf{x}$ then minimises the least-squares error function

$$J(\mathbf{x}) = \sum_{j=1}^{k} (\mathbf{z}(j) - \mathbf{h}(j;\mathbf{x}))^\top N(k)^{-1} (\mathbf{z}(j) - \mathbf{h}(j;\mathbf{x})).$$

Since we are describing an iterative minimization algorithm, we shall assume that we have already obtained an estimate $\hat{\mathbf{x}}^*$ of $\mathbf{x}$. Then a Levenberg-Marquardt update for $\mathbf{x}$ can be computed from the linear system

$$A(\mathbf{x} - \mathbf{x}^*) = \mathbf{a}, \tag{2}$$

where

$$\begin{aligned} A &= \sum_{j=1}^{k} H(j)^\top N(j)^{-1} H(j) + \lambda I, \\ \mathbf{a} &= \sum_{j=1}^{k} H(j)^\top N(j)^{-1} (\mathbf{z}(j) - \mathbf{h}(j;\mathbf{x}^*)) \end{aligned} \tag{3}$$

and $H(j)$ is the Jacobian of $\mathbf{h}(j)$ evaluated at $\mathbf{x}^*$:

$$H(j) = \left. \frac{\partial \mathbf{h}(j)}{\partial \mathbf{x}} \right|_{\mathbf{x}^*}$$

The term λI is a multiple of the identity matrix controlled inside the Levenberg-Marquardt algorithm to resolve conditioning problems. A is then positive-definite, and can be identified as the inverse of the covariance matrix of the state vector, so that A^{-1} is the state covariance. $\mathbf{a}$ is formed from the linearly transformed innovation vectors $\mathbf{z}(j) - \mathbf{h}(j;\mathbf{x}^*)$. Once the modification $\mathbf{x} - \mathbf{x}^*$ has been computed by solving the linear system (2), the new value $\mathbf{x}$ becomes $\mathbf{x}^*$ and another iteration is applied, continuing until convergence is achieved, $\mathbf{a}$ tending to zero.

2.1 Recursive update

After obtaining a batch reconstruction, another image is processed. How can the new data be incorporated? We can augment the state vector $\mathbf{x}$ with the camera position parameters for the new image, and any new structure vectors. We also need to incorporate the new measurements. In general the augmented linear system (2) will take the form

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{12}^\top & A_{22} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 - \mathbf{x}_1^* \\ \mathbf{x}_2 - \mathbf{x}_2^* \end{pmatrix} = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{pmatrix} \tag{4}$$

with the subscript 1 corresponding to the old system, and 2 to the added parameters & measurements. The block A_{12} is not zero in general because the new measurements will involve old parameters, since features are tracked over multiple frames. In this way we can always add new parameters and measurements into the system, and continue to apply Levenberg-Marquardt iterations to the augmented system.

2.1.1 Eliminating state parameters

The obvious problem with this method is that as more data is introduced, the linear system (2) increases in size. To get around this problem, we need to eliminate parameters from the state vector as they become obsolete. We shall consider augmentation and elimination as independent processes to be treated and implemented separately. We call the part of the state vector to be eliminated $\mathbf{x}_1$, and the part to be retained $\mathbf{x}_2$. We assume that no measurements yet to be processed will involve $\mathbf{x}_1$. Also we assume that all the measurements thus far $\mathbf{x}_1$ have been processed to convergence, so that $\mathbf{a}_1 = \mathbf{0}$ and $\mathbf{a}_2 = \mathbf{0}$. The trick is to make sure that as subsequent updates provide results as close as possible as those that would have been obtained if $\mathbf{x}_1$ had been retained in the state vector.

We partition the state vector and information matrix as in (4). We introduce new measurements, which according to the assumptions above do not involve $\mathbf{x}_1$. If we do *not* eliminate $\mathbf{x}_1$, the update follows (4):

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{12}^\top & A_{22} + A' \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 - \mathbf{x}_1^* \\ \mathbf{x}_2 - \mathbf{x}_2^* \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{a}' \end{pmatrix} \tag{5}$$

where A', $\mathbf{a}'$ are the information and innovation terms for the new observations. If we *do* eliminate $\mathbf{x}_1$, we can write down a hypothetical linear system for an update involving only $\mathbf{x}_2$:

$$A(\mathbf{x}_2 - \mathbf{x}_2^*) = \mathbf{a} \tag{6}$$

We want to set A and $\mathbf{a}$ appropriately so that systems (6) and (5) achieve the same result for $\mathbf{x}_2$. Using the standard formula for the inverse of a partitioned matrix, we can reorder (5) to compute the update for $\mathbf{x}_2$ first, and then $\mathbf{x}_1$:

$$\begin{aligned}(A_{22} + A' - A_{12}^\top A_{11}^{-1} A_{12})(\mathbf{x}_2 - \mathbf{x}_2^*) &= \mathbf{a}' \\ A_{11}(\mathbf{x}_1 - \mathbf{x}_1^*) &= -A_{12}(\mathbf{x}_2 - \mathbf{x}_2^*)\end{aligned}$$

Since after elimination, the update for $\mathbf{x}_1$ is not required, we can achieve the desired result by setting

$$A = A_{22} + A' - A_{12}^\top A_{11}^{-1} A_{12}, \qquad \mathbf{a} = \mathbf{a}'.$$

Thus we merely need to subtract $A_{12}^\top A_{11}^{-1} A_{12}$ from the remaining part A_{22} of the information matrix, and then compute subsequent updates as usual. This correction is the Schur complement of A_{11}.

2.1.2 Eliminating observations

We have extended the VSDF to maintain a time window of observations, reducing the effect of non-linearity in the observation model by allowing repeated re-linearisation over all recent observations. Once again, we have to be careful that the results of subsequent updates are as far as possible unchanged by the choice of which observations to eliminate. Writing the vector $\mathbf{a}$ in (3) as

$$\mathbf{a} = \sum_{j=1}^{k} H(j)^\top N(j)^{-1}(\mathbf{z}(j) - \mathbf{h}(j;\mathbf{x}^*)) = \sum_{j=1}^{k} \mathbf{a}(j),$$

let us say, for instance, that once convergence of Levenberg-Marquardt has been achieved, we wish to eliminate all observations up to time $j = k - k_{\text{obs}}$, i.e. retaining a time window of observations of size k_{obs}. Then if we were to perform another L-M iteration, ignoring the eliminated observations, we need to redefine $\mathbf{a}$ as

$$\mathbf{a} = \mathbf{a}_0 + \sum_{j=k-k_{\text{obs}}+1}^{k} \mathbf{a}(j)$$

where $\mathbf{a}_0 = \sum_{j=1}^{k-k_{\text{obs}}} \mathbf{a}(j)$, in order to achieve the same result (no change to $\mathbf{x}$) as if no observations were deleted. Note that if observations are eliminated only when they have been processed to convergence, $\mathbf{a}_0 = \sum_{j=1}^{k} \mathbf{a}(j) = \mathbf{0}$.

To implement this scheme, it is necessary to propagate not only the latest estimate $\mathbf{x}$ and information matrix A, as is normal, but also the vector $\mathbf{a}_0$ for observations eliminated up to the current time. We impose the rule that if state parameters are to be eliminated, any observations involving those parameters should be first processed and iterated to convergence, so that the vector $\mathbf{a}_1$ of innovation terms for the eliminated state parameters is zero, simplifying the process of state parameter elimination.

2.1.3 When to eliminate parameters

It is tempting to eliminate camera motion parameters as soon as the corresponding image is processed. However this is inadvisable because 3D structure vectors need at least two frames to be initialised. Only once a subsequent match for the feature is found, and the structure parameters inserted into the state vector $\mathbf{x}$, can the original observation of the point be processed by Levenberg-Marquardt iteration. Thus future observations can adjust the values of current camera motion estimates, not just through the usual coupling between current and past state estimates, but also through observations being newly processed one or more frames after the observations were obtained. This is the main reason that a time-window of observations and camera motion parameters should be maintained at all times. We allow the size of the time window to be controlled dynamically.

3 Recursive Structure from Motion

We have developed a general-purpose tool for the batch/recursive computation of state parameters which naturally partition into sub-blocks. This tool provides a speedup over conventional least-squares methods when the sensor measurements induce a sparse structure into the state information matrix. In the most general form of the VSDF, described in [12], we show how the VSDF can handle any projection type, and we have implemented 2D/3D projective/affine/Euclidean reconstruction. Indeed information from any sensor can be handled. The method is not restricted to computer vision. However here for the sake of brevity we restrict discussion to 3D Euclidean reconstruction from point features, where the problem is to reconstruct 3D points and camera positions from feature matches (assumed known) in multiple images.

3.1 Batch bundle adjustment

Given n 3D features $\mathbf{X}_i$, $i = i = 1, \ldots, n$, and k camera positions described by rotations $R(j)$ and translation vectors $\mathbf{T}(j)$, $j = 1, \ldots, k$, we can write the projection of the point onto the image in homogeneous coordinates, assuming no non-linear image distortion, as

$$\begin{pmatrix} \mathbf{z}_i(j) \\ 1 \end{pmatrix} = \lambda K R(j)(\mathbf{X}_i - \mathbf{T}(j))$$

where K is the 3×3 matrix of camera calibration parameters, here assumed known. Eliminating the scale factor λ, we can write this as

$$\mathbf{z}_i(j) = \mathbf{h}_P(R(j), \mathbf{T}(j), \mathbf{X}_i) + \mathbf{w}_i(j) \tag{7}$$

where $\mathbf{h}_P$ is the projection function for 3D points, and $\mathbf{w}_i(j)$ is a random noise vector with covariance $N_i(j)$. Assuming that approximate values for all the parameters to be estimated are available or may be computed, the standard geometric error optimization approach minimizes the differences between the feature projections defined by equation (7) and the actual image feature locations, over all available images and corresponding matched features. The parameters adjusted are the 3D structure vectors $\mathbf{X}_i$ and the camera locations $R_{(j)}$, $\mathbf{T}_{(j)}$.

We firstly stack up all the unknown motion and structure parameters into a single vector of state parameters $\mathbf{x}$. Writing the structure parameters for a single 3D point $\mathbf{X}_i = (X_i \; Y_i \; Z_i)^\top$ as $\mathbf{x}_i = \mathbf{X}_i$, and the motion parameters for a single camera position as $\mathbf{x}_{(j)} = (\mathbf{r}_{(j)}{}^\top \; \mathbf{T}_{(j)}{}^\top)^\top$, where $\mathbf{r}_{(j)}$ represents the rotation $R_{(j)}$ in some way such as Euler angles or (better) quaternions or local exponential coordinates, we write the complete state vector for the k images of n 3D features as

$$\mathbf{x}^\top = (\mathbf{x}_1^\top \; \ldots \; \mathbf{x}_n^\top \mid \mathbf{x}_{(1)}{}^\top \; \ldots \; \mathbf{x}_{(k)}{}^\top). \tag{8}$$

If we also stack all the measurements into a single measurement vector $\mathbf{z}$ and projection function $\mathbf{h}(.)$, with associated stacked observation covariance matrix N, then A, $\mathbf{a}$ are computed as

$$A = H^\top N^{-1} H, \qquad \mathbf{a} = H^\top N^{-1} (\mathbf{z} - \mathbf{h}(\mathbf{x}))$$

It has long been known in the photogrammetry community, in the form of the equivalent *normal equation* formulation, that the matrix A in the adjustment formula (2) takes a special sparse form in the context of reconstruction, that allows the use of sparse matrix methods to greatly accelerate the computation of the adjustment [20]. Assuming that each image measurement involves only a single image and 3D feature, we can obtain the following shape for A:

$$A = \left(\begin{array}{ccc|ccc} A_{11} & 0 & . & A_{1(1)} & \ldots & A_{1(k)} \\ 0 & . & 0 & : & . & : \\ . & 0 & A_{nn} & A_{n(1)} & \ldots & A_{n(k)} \\ \hline A_{1(1)}{}^\top & \ldots & A_{n(1)}{}^\top & A_{(11)} & 0 & . \\ : & . & : & 0 & . & 0 \\ A_{1(k)}{}^\top & \ldots & A_{n(k)}{}^\top & . & 0 & A_{(kk)} \end{array}\right) \tag{9}$$

The block-diagonal quadrants at top-left and bottom-right of (9) signify that the inverse of the matrix A may be computed quickly, in fact in time proportional to either n or k. This is achieved by the recursive partitioning method, either (i) eliminating the structure blocks from the linear system (2), solving for the motion parameter adjustments, and back-substituting to get the structure adjustments, or (ii) vice versa. If camera calibration is required then A is augmented by an extra set of blocks for the camera calibration parameters $\mathbf{x}_{\text{cal}}$.

In practice then, batch updates of structure and motion can be computed in time linear with the number of structure vectors n. We would like if possible to maintain this efficiency in recursive updates. Now applying the recursive methods of section (2), we see that to eliminate structure vector $\mathbf{X}_1$, for example, correcting the remaining blocks of A by computing the Schur complement of A_{11} results in the corrections

$$A_{(j_1 j_2)} \leftarrow A_{1(j_1)}{}^\top A_{11}^{-1} A_{1(j_1)}, \quad j_1, j_2 = 1, \ldots, k.$$

Thus eliminating the structure vectors fills in all the motion information blocks of A. However the structure part of A remains sparse. To see the effect of eliminating motion parameters, for example $\mathbf{x}_{(1)}$, we firstly re-order the blocks of A in (9) so that the row and column corresponding to $\mathbf{x}_{(j)}$ are moved to the top and left of A, resulting in the matrix

$$A = \left(\begin{array}{c|ccc|ccc} A_{(11)} & A_{1(1)}{}^\top & \ldots & A_{n(1)}{}^\top & 0 & \ldots & 0 \\ \hline A_{1(1)} & A_{11} & 0 & . & A_{1(2)} & \ldots & A_{1(k)} \\ : & 0 & . & 0 & : & . & : \\ A_{n(1)} & . & 0 & A_{nn} & A_{n(2)} & \ldots & A_{n(k)} \\ \hline 0 & A_{1(2)}{}^\top & \ldots & A_{n(2)}{}^\top & A_{(22)} & 0 & . \\ : & : & . & : & 0 & . & 0 \\ 0 & A_{1(k)}{}^\top & \ldots & A_{n(k)}{}^\top & . & 0 & A_{(kk)} \end{array}\right)$$

Then we split off the diagonal block $A_{(11)}$ and correct the remainder of A using the Schur complement of $A_{(11)}$, resulting in the adjustments

$$A_{i_1 i_2} \leftarrow A_{i_1 i_2} - A_{i_1(1)} A_{(11)}{}^{-1} A_{i_2(1)}{}^\top$$

So eliminating motion fills in the structure blocks. This has to be avoided to maintain update times proportional to n. So our *partial elimination adjustment* method is to ignore corrections that fill-in zero blocks, while applying the correction to the blocks which are already non-zero.

4 Results

We first present results on real images. Figure 1 shows four images from a sequence of ten, with the features matched from frame to frame superimposed. The mobile robot moved across the scene, sideways relative to the camera, in steps of around 20cm. The feature matching was obtained using the method described in [19], which uses camera motion estimates obtained from robot odometry to drive junction and line matching using the epipolar constraint between each pair of views. Points and lines were integrated in the matching and the reconstruction. Figure 2 shows the result of Euclidean reconstruction, Planes are reconstructed along with the points and lines, and as more views are processed, more surfaces come into view. This allows us to texture map the reconstructed shape. The plane on the top of the filing cabinet is recovered incorrectly, due to the combination of imperfect camera motion estimates from odometry and near-horizontal lines, whose recovery in 3D from a horizontal camera motion is problematic.

We wish to test the effect of our improvements to the recursive reconstruction algorithm presented in [8]. We generated a simple scene of 20 points arranged at random within a sphere, and created a sequence of 50 images by moving the camera through a half-circle around the sphere of points, rotating the camera to keep the points in view.

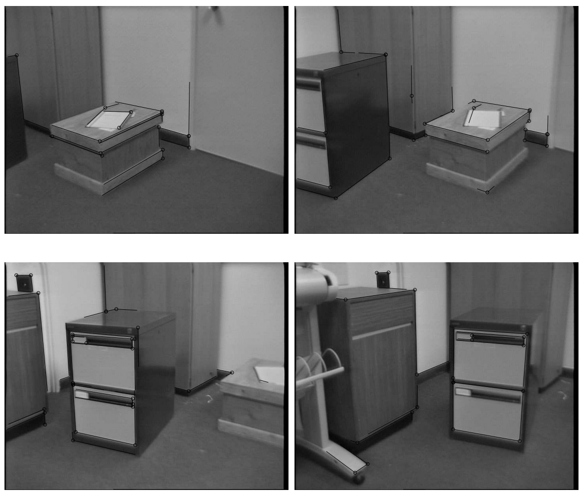

Figure 1. Images 1, 4, 7 and 10 from the sequence of 10 used to test the recursive algorithm.

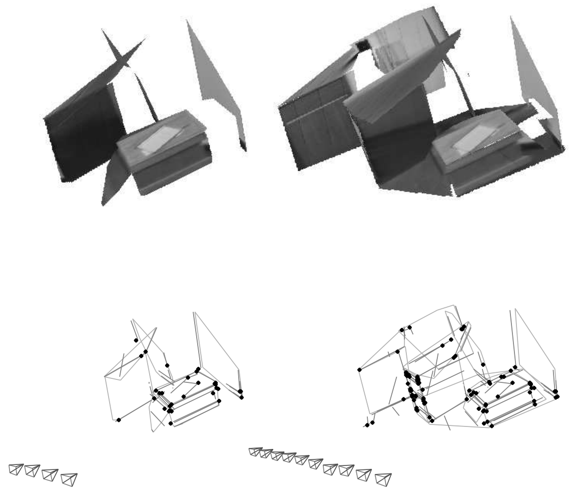

Figure 2. The development of the recursive reconstruction, starting with a batch process from the first four images. The recovered points and lines are shown in the lower images, along with the camera locations and viewing directions. The upper images show the textured planes recovered.

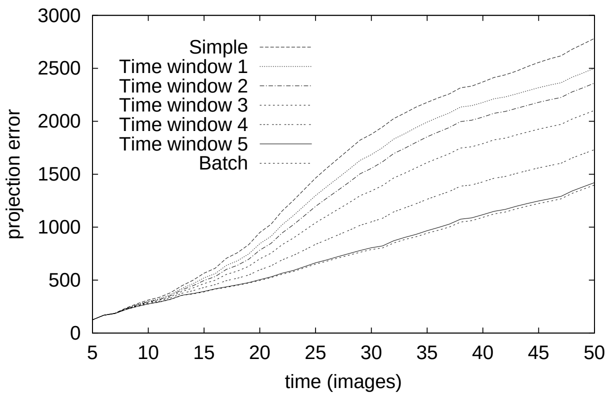

Figure 3. Comparison of the performance of recursive reconstruction on simulated data for different sizes of time window $k_{\rm obs}$.

The distance of the camera to the points is 5000 units, and the sphere has radius 1800 units. The focal lengths of the camera were set at 320 and 380 pixels in the horizontal and vertical directions respectively, and the image centres were set in the centre of the 256×256 pixel images. Gaussian noise with standard deviation 0.1 pixels was added to each image point. To test the performance of our algorithm with incomplete data, 15% of the projected image features were discarded as "invisible". Batch processing was applied over the first five images, and then recursive processing for the remainder of the sequence. After each image was processed, the total squared image-plane error was measured using the latest estimates of structure and motion. We ran the algorithm for different values of time window $k_{\rm obs}$, which indicates both the number of frames of observations retained and the number of frames of motion parameters. The results are shown in figure 3. Two additional graphs show the result of batch processing over each sub-sequence (obtained by setting $k_{\rm obs} = 50$), and the "simple" method obtained by eliminating the motion parameters completely from the updates, so that motion and structure are computed separately, as in [7, 17]. We see that as the size of the time window $k_{\rm obs}$ is increased, the results approach the accuracy of the batch method.

To see the effects of varying the type of information adjustment, figure 1 compares the three modes of information adjustment upon elimination of motion and/or structure parameters: **None**: apply no compensation. With a time window of one frame this is the "VSDF" mode in [8], unless the motion parameters are completely eliminated and computed separately, in which case we get the "simple" case from [8]; **Partial** adjustment: adjust blocks without allowing fill-in. This is the procedure we recommend; and **Full** adjustment: adjust the information matrix with fill-in as necessary. With

	Time window				
Adjustment	1	3	5	7	9
None	2007	1822	1431	1402	1402
Partial	2500	2103	1420	1400	1399
Full	1401	1399	1398	1398	1398

Table 1. Table of final total error residuals for different time window values k_{obs} and different information compensation modes.

a time window of one frame this is the "linear-optimal" method in [8]. We see that for this data-set, for time windows up to three frames, the "None" mode works best in terms of minimising the error, whereas for larger time windows the partial adjstment is better. We would recommend using a time window of around five frames with data as fragmentary as this.

5 Conclusions

We have presented a new information matrix adjustment technique to the recursive least-squares problem, in the case that parameters may be introduced and eliminated at any time. Elimination proves to be the stumbling block because it causes fill-in of the information matrix, slowing subsequent computations. We have proposed a compromise adjustment method without fill-in, and shown that it can improve the results.

The other innovation is the introduction of a time window of retained motion parameters and observations. We believe that without this feature recursive approaches to 3D scene and motion reconstruction cannot hope to achieve accuracy approaching that of an optimal batch algorithm. In our results increasing the time window yielded a significant improvement in accuracy, at the cost of somewhat reduced speed.

References

[1] A. Azarbayejani and A. P. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):562–575, 1995.

[2] T. Broida, S. Chandrashekhar, and R. Chellappa. Recursive estimation of 3-D kinematics and structure from a long image sequence. *IEEE Transations, AES*, 26(4):639–656, 1990.

[3] A. Chiuso, P. Favaro, H. Jin, and S. Soatto. Mfm: 3-D motion and structure causally integrated over time: Part ii: Implementation. In *Proc. 6th European Conf. on Computer Vision, Dublin*, 2000.

[4] N. Cui, J. Weng, and P. Cohen. Recursive-batch estimation of motion and structure from monocular image sequences. *Computer Vision, Graphics, and Image Processing*, 59(2):154–170, 1993.

[5] A. Davison and D. Murray. Mobile robot localisation using active vision. In *Proc. 5th European Conf. on Computer Vision, Freiburg*. Springer-Verlag, June 1998.

[6] A. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *Proc. 5th European Conf. on Computer Vision, Freiburg*, volume 1, pages 311–326. Springer-Verlag, June 1998.

[7] C. Harris and J. M. Pike. 3D positional integration from image sequences. In *Proc. 3rd Alvey Vision Conf., Cambridge*, pages 233–236, 1987.

[8] P. McLauchlan and D. Murray. A unifying framework for structure and motion recovery from image sequences. In *Proc. 5th Int'l Conf. on Computer Vision, Boston*, pages 314–320, June 1995.

[9] P. McLauchlan and D. Murray. Active camera calibration for a Head-Eye platform using the variable State-Dimension filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):15–22, 1996.

[10] P. F. McLauchlan. Gauge independence in optimization algorithms for 3d vision. In *Proc. ICCV'99 Vision Algorithms Workshop*, 1999.

[11] P. F. McLauchlan. Gauge invariance in projective 3d reconstruction. In *IEEE Workshop on Multi-View Modeling and Analysis of Visual Scenes, Fort Collins, CO, June 1999*, 1999.

[12] P. F. McLauchlan. The variable state dimension filter. Technical Report VSSP 4/99, University of Surrey, Dept of Electrical Engineering, Dec. 1999.

[13] P. F. McLauchlan and A. Jaenicke. Accurate mosaicing using structure-from-motion methods. Technical Report VSSP 5/99, University of Surrey, Dept of Electrical Engineering, Nov. 1999.

[14] P. F. McLauchlan and J. Malik. Vision for longitudinal control. In A. Clark, editor, *Proc. 8th British Machine Vision Conf., Essex*. BMVA Press, 1997.

[15] P. F. McLauchlan, I. D. Reid, and D. W. Murray. Recursive affine structure and motion from image sequences. In *Proc. 3rd European Conf. on Computer Vision, Stockholm*, volume 1, pages 217–224. Springer-Verlag, May 1994.

[16] P. F. McLauchlan, X. Shen, P. Palmer, A. Manessis, and A. Hilton. Surface-based structure-from-motion using feature groupings. To be presented at ACCV, Taiwan, 2000.

[17] P.A.Beardsley, A.Zisserman, and D.W.Murray. Sequential updating of projective and affine structure from motion. *International Journal of Computer Vision*, 23(3), 1997.

[18] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.

[19] X. Shen and P. Palmer. Uncertainty propagation and the matching of junctins as feature groupings. Submitted to IEEE PAMI, 1999.

[20] C. Slama, C. Theurer, and S. Henriksen, editors. *Manual of Photogrammetry*. American Society of Photogrammetry, 1980.